FERMILAB-Conf-92/308

The DØ Software Trigger

James T. Linnemann
for the DØ Collaboration

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

Michigan State University, Department of Physics and Astronomy
East Lansing, Michigan 48824

October 1992

Submitted to Proceedings of *Computing in High Energy Physics 1992,*
Annecy, France, September 1992

The DØ Software Trigger

James T. Linnemann, Fermilab and Michigan State University

for the DØ Collaboration [1]

In the DØ experiment, the software filter operates in a processor farm with each node processing a single event. Processing is data-driven: the filter does local processing to verify the candidates from the hardware trigger. The filter code consists of independent pieces called "tools"; processing for a given hardware bit is a "script" invoking one or more "tools" sequentially. An offline simulator drives the same code with the same configuration files, running on real or simulated data. Online tests use farm nodes parasiting on the data stream. We discuss the performance of the system and how we attempt to verify its correctness.

Design

In the DØ experiment data collection [1] begins with a level 1 trigger. The data is digitized and flows into multiported memories of a node in the level 2 (software) farm. The level 2 farm consists of 32 filtering nodes, a supervisor, and a monitor node. Each filtering node has a VAX 4000/60 15 MIP processor with 8MB of main memory (we currently need 4MB) and 8 multiported raw data memories.

The filtering nodes have access to complete events. However, the processing performed in the software trigger is data driven. The level 1 trigger provides a mask of 32 bits which summarizes its decision. Candidate objects found at this level are part of the event data, and are each tagged with the level 1 bits to which they are relevant. The level 1 bits determine what software filtering is done. We define 128 level 2 bits, which are on if the corresponding filter was run and passed. These level 2 bits steer data to recording streams and monitoring tasks.

Configuration files describe the level 1 trigger and level 2 triggers. The configuration files are used both for actually running the experiment and for driving the simulators. The files are easier to read than to write, so we use a tool (TRIGPARSE, written in LISP) which generates the configuration files from a higher-level description, and allows for default values. We typically run with 25-30 Level 1 bits, and 50-60 level 2 bits defined.

The processing for each level 2 bit is called a "script". A script consists of a list of filter "tools" to apply, and the cuts to use for this particular invocation. The tools are elementary filter processes such as em clusters, muons, jets, and missing Et. The tools are applied sequentially within a script; the script fails when the first tool of a script fails.

The tools have 3 entry points: INIT, to build the constants to download; PARAMS, to receive the list of potential cuts; RUN, to perform the filtering, given the Level 1 bit

[1] D0 instituitions are : Los Andes, Arizona, BNL, Brown, U.C. Riverside, CBPF, CINVESTAV, Columbia, Delhi, FNAL, Florida State, Hawaii, Illinois, Indiana, Iowa State, LBL, Maryland, Michigan, Michigan State, Moscow State, NYU, Northeastern, N. Illinois, Northwestern, Notre Dame, Panjab, IHEP-Protvino, Purdue, Rice, Rochester, CEN Saclay, Stony Brook, SSCL, Tata Institute, Texas Arlington, Texas A & M.

number and the cut set to use. These tools do localized processing on the relevant Level 1 candidates, and remember previous processing of an event. Results are recorded in both a general summary bank ESUM, and in more specialized banks. The total amount of information generated by filter processing is roughly 10% of the total raw event size.

The ESUM (Event SUMmary) banks are very useful. An ESUM bank consists of a list of 4-vectors and particle ID's for each object found by the filters. A linear chain of ESUM banks is produced, with an origin tag for Level 1, Level 2, full reconstruction, Monte Carlo, etc. We use the ESUM bank in monitoring programs, for rate calculations, and as a means of communicating between tools (eg for higher-level tools which calculate masses or topological cuts). We built a generic comparison tool which compares objects of a given type between two ESUM banks of different origins and have used it in comparing online vs offline level 2 results, level 2 vs MC inputs, and level 2 vs full reconstruction. The comparison with full reconstruction is also done on real data taken in a mark-and-pass mode in which filter results are written but no rejection is performed.

The level 2 simulator is driven by the standard configuration files. The only difference with the online code is the mechanism for obtaining the data (standard Zebra I/O instead of the EPASCAL realtime code). The simulator is a package like any other in our standard offline frameworks and can be combined with any offline code. All of our code is tested offline with this simulator before it is downloaded. The trigger simulation was used in studies for trigger strategy and rate estimation on MC signal and QCD events.

Development and Performance

Our code development generally followed standard DØ offline portable FORTRAN guidelines, except that filter code is not allowed to do I/O other than calls to our standard error message utilities. The bulk of the specialized design effort was in fast localized unpacking for the calorimeter data, including attention to organization of the raw data. Both the em cluster and jet finder were developed separately from the offline algorithms. The more complex muon code is mostly a subset of the offline code, with restrictions to regions of interest and special cosmic ray rejection code. Considerable effort has gone into tuning parameters based on test beam data, and comparing the online algorithms against the offline algorithms. The overall effort could be summarized by saying there were essentially no problems found particular to running the code in the filter nodes (under the ELN operating system) other than upgrading the resistance to bad raw data, and verifying that download of the cuts and constants succeeded in all nodes. We were able to verify online filter results bit for bit with the simulator.

The time budget for filtering can be deduced from the number of nodes available and input rate from the Level 1 trigger. Presently we operate at roughly 25 nodes / 50 Hz input = 500 ms/event allowed. With upgrades in the number of processing nodes and in the data collection hardware we expect this to drop to perhaps 50 nodes / 150 Hz = 300 ms. We presently spend 100 ms per event in verifying the hardware checksum, but expect this will become unneccessary soon. We are currently taking 150-200 ms/event filtering, less than our design time of 250 ms/event in the filters, and rejecting by about a factor of 30. Our Level 1 rate is currently limited by long tails in the processing time in the filtering nodes (rms of about 400 ms); we keep the deadtime introduced by the filter farm below 5%. We hope to improve this by even more localized treatment of muon data. Other projects include better treatment of hot cells in missing pt, and adding a track match requirement for electrons.

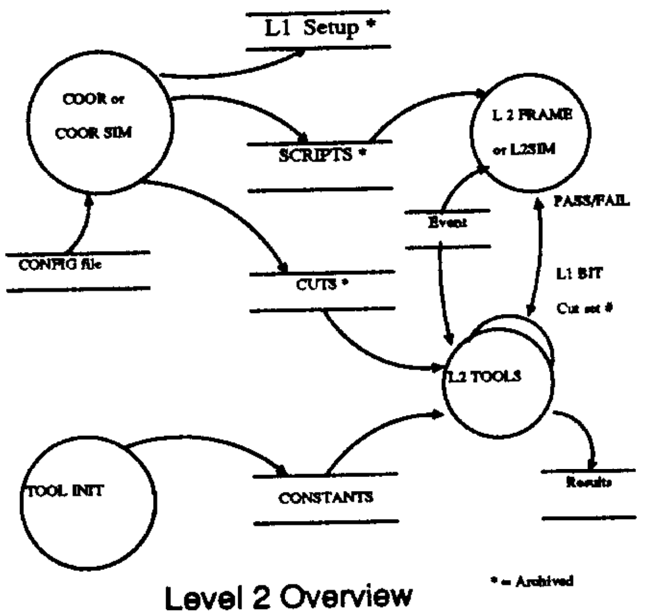
Monitoring and Testing

We monitor the system by displays showing in color the state of each node and the number of nodes ready for a new event; it updates every 10 seconds. Error messages from the filter code are directed to the central alarm handler. We can interactively view pass rates and timing distributions. Monitoring tasks make histograms from the filter result banks. The ELN remote debugger and a tool for dumping events in a node are used for understanding crashes, which are now fairly infrequent, perhaps one every few days. We have a FZBROWSE utility for examining our download files and event dumps.

When we make new releases, we compare with results on standard baseline event samples. We are beginning to exploit a "shadow" mode of operation in which a few filtering nodes are for online checkout of new code before running in the regular nodes. This mode generates no level 2 deadtime and parasites off the level 1 bits defined for the regular run, since the level 1 bits are a scarce resource. We are experimenting with making histograms inside the level 2 nodes, where it is possible to gather higher statistics than possible in monitoring of only events which pass the filters.

References

[1] See also the talks: Dave Cutts, *Operation of the D0 Data Acquisition System*; Bruce Gibbard, *Run Control and Resource Management in the D0 Run Time System.*



Level 2 Overview